

Remarks/Arguments

With reference to the Office Action mailed February 4, 2003, Applicants offer the following remarks and argument.

Status of the Claims

Claims 1-16 were originally presented for examination. All sixteen of the claims were rejected in the Office Action mailed May 21, 2004. Applicants have extensively amended the claims to positively recite the features shown in Figures 2 and 3, and described in numbered paragraphs 78-93.

The Office Action of May 21, 2004

Claims 1-6 were rejected under 35 USC §102(e) as being anticipated by United States Published Patent Application 20020032709A1 to Gessner for “Network Client Accepts And Processes Replaceable Document Type Definition Components Containing Corresponding Grammars And Transforms Documents According The Same.”

As to claim 1, ¶ 25’s use of the phrase “fairly simple scanner, , ¶ 29’s use of the phrase “now allows such tokens and constructs to be fixed”, , ¶ 28’s disclosure of DTD’s, and ¶ 29’s use of the <p>...<p> example, taken together are said to anticipate independent claim 1.

As to claim 2 the Office Action relies on Gessner’s disclosure at , ¶ 27 of parser component 104, at , ¶ 29 of the parsing engine, and again at , ¶ 29 of “well formed expressions.”

As to claim 3, the Office Action relies on the disclosures at , ¶28 and , ¶29 of the parser and replaceable DTD’s.

As to claim 4, which claims

“4. The parsing system according to claim 3, further comprising:

corresponding information storage means for storing information that correlates the type of data string with a syntax recovery unit for recovering from an error in said data string,

wherein, in accordance with the type of target data string, said parser employs said information stored in said corresponding information storage means to set up said syntax recovery unit for the correction of an error upon the receipt of a request.”

The Office Action relies upon paragraphs 33 and 41,

“[0033] Referring now to FIG. 2, depicted therein is a system diagram of an automatic data processing system 200 that includes a processor arrangement 202 including one or more processing elements such as CPU's (Central Processing units), a data storage subsystem 204 such as one including multiple disk-based data storage facilities, and an I/O subsystem 206 to facilitate network communications, etc. A network client, such as one including parsing engine 100, may be expected to facilitate the receipt and manifestation of content in accordance with particular grammars contained within particular replaceable DTD components 106. Automatic data processing system 200 facilitates the loading and execution of a browser environment, for example, that may manifest and display HTML, etc. content in accordance with particular grammars contained within DTD components 106.

and

“[0041] Next, at step S3-3, one or more DTD components containing particular grammars relative to data streams to be rendered with in a network client environment, are acquired via a network connection. Of course such DTD components may be locally stored, but the present invention does contemplate the notion that such DTD components along with corresponding documents formatted according to corresponding grammars may be stored remotely on server systems and delivered at and during run time of a browser, etc. to facilitate dynamic replacement of particular grammars and to further facilitate the rendering of content based thereon.”

To anticipate the first clause of claim 4, and on paragraph 35¹,

“[0035] In implementing and operating parsing engines such as parsing engine 100, several phases of operation occur. In particular, in a first phase, object construction occurs during the parsing of a document. The containing application (e.g., a network client), initiates the parse by creating a URL tokenizer object, and an HTML parse object. The parser is assigned a sink such as sink component 108, and a DTD component such as one from replaceable DTD components 106. The DTD component selected understands the grammar of the document being parsed, while the sink component interfaces to allow the DTD component to properly build a content model for later or subsequent rendering, layout, or manifestation via appropriate and well known document rendition modules, objects, processes.”

¹ corresponding information storage means for storing information that correlates the type of data string with a syntax recovery unit for recovering from an error in said data string,

To anticipate the second clause of claim 4².

The Office Action relies upon ¶ 37, which recites

“[0037] A third phase of operation involves tokenization. The tokenizer aspect of parser component 104 controls and coordinates the tokenization of the input stream into a collection of tokens. Different grammars will have their own subclasses of tokens as well as their own corresponding DTD components. As the tokenizer runs, it repeatedly calls methods to get additional tokens for processing. Such tokenization iteration continues until an end-of-file occurs on an input stream, an unrecoverable error occurs such as network traffic stoppages or delays, etc.”

to anticipate claim 5.

Claim 6, which recites

“6. The parsing system according to claim 2, further comprising: a lexical analyzer, for performing token analysis for said target data string; and a token recovery unit, for correcting an error detected by said lexical analyzer in said token in said data string, wherein said token recovery unit can change the contents of a correction.”

Is said to be anticipated by paragraphs 25 and 29,

“[0025] Now with specific attention to the components within parsing engine 100, the scanner component 102 is the first major component. Scanner component 102 provides an incremental "push-based" API (application program interface) that provides methods for accessing characters in an input stream (usually a URL, a uniform resource locator), finding particular sequences, collating input data, and skipping over unwanted data. A fairly simple scanner can be implemented and used effectively to parse everything from HTML and XML to C++. Such a scanner and the implementation thereof will be readily understood by those skilled in the art.

“[0029] The present invention may be configured to allow for malformed documents and expressions to be constructed based on corresponding grammars contained within DTD components. That is, the present invention allows dynamic configuration of parsing systems through use of replaceable DTD components 106, and the transformation of otherwise malformed documents and expressions into well-formed expressions that a content model, for example, can later understand and process. Such transformation processes may be borrowed from artificial intelligence processing schemes and may involve rules of propagation, etc. For example suppose a parsing engine 100 were to realize a token for the start of paragraph text outside of an HTML file starting tag. Such a construct would otherwise be inappropriate and, in some cases by some browsers, unrenderable and therefore discarded. In contrast, the present invention now allows such tokens and constructs to be "fixed" or corrected (i.e., transformed into well-formed expressions) based on document context as defined by replaceable DTD components provided by the present invention, etc. That is, by recognizing the context of the tokens realized by a parsing engine

² wherein, in accordance with the type of target data string, said parser employs said information stored in said corresponding information storage means to set up said syntax recovery unit for the correction of an error upon the receipt of a request.”

provided by the present invention, etc., the present invention can transform otherwise malformed expressions and documents into well-formed objects which can be processed by an appropriate content model.

Claim 7 which recites “multiple types of said token recovery units are prepared in accordance with the type of error that is detected by said lexical analyzer in said data string, and each has a function for correcting a specific type of error.” Is said to be anticipated by Gessner’s paragraphs 28-29³.

Previously applied numbered paragraphs 25, 27, 28, and 29 are said to anticipate independent claim 8.

Previously applied numbered paragraphs 25 and 29 are said to anticipate independent claim 9.

³ [0028] In parsing engine 100, one or more replaceable DTD components 106 may be utilized. Such DTD components 106 describe the rules for well-formed and/or valid documents in a target grammar and, more particularly, for well-formed expressions and objects within a particular grammar. For example, in HTML, the DTD declares and defines the tag sets, the associated set of attributes, and the hierarchical (nesting) rules of HTML tags. That is, a DTD component according to the present invention will declare, in the case of HTML for example, a tag set expression for paragraph text (e.g., “<p> . . . text stream . . . <p>”). Other expressions will be defined by such a grammar construct as provided by DTD components. Such definitions will be immediately understood by those skilled in the art. Once again, by separating the DTD components 106 from other components in the parser engine, it becomes possible to use the same system to parse a much wider range of document types and those containing expressions corresponding to different and varying rules of grammar. Simply put, this means that the same parser can provide an input to the browser biased (via the DTD components 106) to behave like any other HTML browser. The same can be said for XML, etc.

[0029] The present invention may be configured to allow for malformed documents and expressions to be constructed based on corresponding grammars contained within DTD components. That is, the present invention allows dynamic configuration of parsing systems through use of replaceable DTD components 106, and the transformation of otherwise malformed documents and expressions into well-formed expressions that a content model, for example, can later understand and process. Such transformation processes may be borrowed from artificial intelligence processing schemes and may involve rules of propagation, etc. For example suppose a parsing engine 100 were to realize a token for the start of paragraph text outside of an HTML file starting tag. Such a construct would otherwise be inappropriate and, in some cases by some browsers, unrenderable and therefore discarded. In contrast, the present invention now allows such tokens and constructs to be “fixed” or corrected (i.e., transformed into well-formed expressions) based on document context as defined by replaceable DTD components provided by the present invention, etc. That is, by recognizing the context of the tokens realized by a parsing engine provided by the present invention, etc., the present invention can transform otherwise malformed expressions and documents into well-formed objects which can be processed by an appropriate content model.

As to independent claim 10, the input unit is said to be anticipated by Figure 1's showing of a scanner component, the second clause reciting a processor and a function is said to be anticipated by numbered paragraph 33, the elements of the processor are said to be anticipated by numbered paragraphs 25 and 29, and the recovery units are said to be anticipated by numbered paragraphs 29, 26, and 37.

Numbered paragraphs 25 and 29 are applied against claim 11's recitation of parsing means and syntax recovery means.

Numbered paragraphs 27 and 29 are applied to independent claim 12's recitation of selecting a program module to correct an error in a target data string in accordance with a syntax rule, parsing the string, issuing a correction request, and correcting the error.

Numbered paragraphs 27 and 28 are applied against claim 13's recitation examining the type of target data string and employing the type of target data string to select an appropriate program module.

Numbered paragraph 31⁴ is applied to claim 14's recitation of

"The parsing method according to claim 12, further comprising the step of:

replacing, upon the receipt of an instruction from said program module to which said correction request has been issued, a rule used for said parsing with a different rule,

wherein, at said step of performing said parsing for the resultant data string, said parsing is performed for said data string written in accordance with said different rule."

Claims 15 and 16 are rejected as anticipated by numbered paragraphs 25 and 27-29.

⁴ [0031] The components shown in FIG. 1 as part of parsing engine 100 may be implemented as language parsing and translation routines within a object oriented programming environment such as within C++, etc. It is important to note, that replaceable DTD components 106 are, in fact, replaceable at run time. That is, because replaceable DTD components 106 may be replaced with other DTD components corresponding to other document type and parsing grammars, a network client such as a web browser including the same may operate upon content received via a network connection that is formatted (well formed, preferably) and manifested within a web browser environment based upon a particular set of grammars and rules.

The Art of Record

The sole reference is United States Published Patent Application 20020032709A1 to Gessner for “Network Client Accepts And Processes Replaceable Document Type Definition Components Containing Corresponding Grammars And Transforms Documents According The Same.”

Gessner describes a method and system for receiving and extracting “renderable” content including “replaceable document type definitions” that are used to facilitate rendering. Specifically, Gessner describes a network client, for example, a world wide web browser and a corresponding method that includes and utilizes a scanner. The scanner accesses the input content stream to extract its renderable content. Gessner next describes a parsing component coupled to the scanner component. The parser component parses the renderable content. Gessner also describes replaceable document type definition components. This is configured to control the parsing component based on a particular document type definition corresponding to a particular grammar. The “replaceable document type definition component” is disclosed by Gessner to be “replaceable during execution of the network client.”

Specifically, Gessner describes a need for a web browser or network client that can be dynamically altered, for example, during runtime, to facilitate the receipt, interpretation, processing, and manifestation of content and data formatted according to a document type definition that is not known a priori. Gessner describes that this need is met by the method and system that processes a received document within a browser environment based on a parsing grammar that is dynamically received and inserted into a parsing engine.

Gessner disclosed that this is accomplished by combining computer language parsing and processing techniques with browser and network client technologies to process

documents formatted based on grammars that are otherwise not known a priori to runtime.

Gessner describes the application of a scanner component that accesses an input content stream via a network connection to extract renderable content with a parsing component. The parsing component is coupled to the scanner component for parsing the renderable content. Included is a replaceable document type definition component that is configured to control the parsing component based on a particular document type definition corresponding to a particular parsing grammar. The replaceable document type definition component is replaceable during runtime.

Applicant's Claimed Invention

Applicants have amended the claims to positively point out and recite their invention. Specifically, the claims now recite a method and system comprising:

1. Analysis means for analyzing the structure of a data string written in accordance with a predetermined rule. The rule provides for:
 - a) generating a grammar information object;
 - b) emptying buffers to receive a data string;
 - c) causing a context pointer to indicate a root node of an abstract syntax tree;
 - d) obtaining a set of syntax recovery units;
 - e) inputting a data string comprising a stream of tokens to the buffers;
 - f) extracting a token from the buffer as a target token;

The rule provides that when the target token is the end of the data string an abstract data tree is generated and an abstract data tree is output.

2. Error detection means for detecting an error in accordance with said predetermined rule by a method that further comprises receiving a non-terminal token, determining from the grammar information object if the token matches a context pointer, and if so adding the token to the context pointer. This is accomplished by the steps of:

- a) adding a node to which the terminal token is appended to the context pointer;
 - b) shifting the point indicated by the by the context pointer to the newly added node to which the non-terminal token is appended;
 - c) repeating steps a) and b) for all child nodes;
 - d) shifting the context pointer to the parent node; and
 - e) obtaining a next token if the node does not contain all child nodes or if the context pointer points back to a parent node.
3. Error recovery means, which, upon the receipt of a request from the analysis means, corrects the apparent error in accordance with the predetermined rule. The error recovery means includes a set of syntax recovery units that individually employ simple functions for correcting specific types of errors. The recovery means selectively employs the syntax recovery means based on the error type in accordance with predetermined rule in order to correct a variety of errors in said data string by a method that comprises sequentially using syntax recovery units where a syntax recovery unit receives an error message if a token does not grammatically match the syntax pointer, and the syntax recovery unit sequentially passes the error to a subsequent syntax recovery unit until a syntax recovery unit matching the error type is found, said syntax recovery unit
- a) resets the current abstract syntax tree, buffer, and context pointer;
 - b) recovers a pointer where in accordance with said syntax rules no error exists; and
 - c) notifies the parser that the correction was successfully applied.

Discussion

The overarching issue presented is whether the sole reference, United States Published Patent Application 20020032709A1 to Gessner for “Network Client Accepts And Processes Replaceable Document Type Definition Components Containing

Corresponding Grammars And Transforms Documents According The Same” in any way teaches or suggests Applicants’ claimed invention.

Specifically, Gessner fails to teach or suggest Applicants’ claimed combination of:

1. Analysis means for analyzing the structure of a data string written in accordance with a predetermined rule.
2. Error detection means for detecting an error (in the content and not the transport).
3. Error recovery means, which corrects the detected error in accordance with a predetermined rule.

Turning specifically to the claim language and Gessner, it is clear that Gessner fails as a reference.

1. Applicants’ claims recite analysis means for analyzing the structure of a data string written in accordance with a predetermined rule, where the rules include:
 - a) generating a grammar information object;
 - b) emptying buffers to receive a data string;
 - c) causing a context pointer to indicate a root node of an abstract syntax tree;
 - d) obtaining a set of syntax recovery units;
 - e) inputting a data string comprising a stream of tokens to the buffers;
 - f) extracting a token from the buffer as a target token; and
 - g) generating an abstract data tree when the target token is the end of the data string.

The nearest that Gessner discloses to Applicants’ “analysis means” are the document type definition objections (104) and their application to the parsing engine (100). While extensively described in Paragraph 28⁵ and 29⁶ of Gessner, the description is not anticipatory of Applicant’s “analysis means.”

⁵ [0028] In parsing engine 100, one or more replaceable DTD components 106 may be utilized. Such DTD components 106 describe the rules for well-formed and/or valid documents in a target grammar and, more particularly, for well-formed expressions and objects within a particular grammar. For example, in HTML, the DTD declares and defines the tag sets, the

2. Error detection means for detecting an error in accordance with said predetermined rule by a method that further comprises receiving a non-terminal token, determining from the grammar information object if the token matches a context pointer. More particularly, Gessner fails error detection by the steps of:
 - a) adding a node to which the terminal token is appended to the context pointer;
 - b) shifting the point indicated by the by the context pointer to the newly added node to which the non-terminal token is appended;
 - c) repeating steps a) and b) for all child nodes;
 - d) shifting the context pointer to the parent node; and
 - e) obtaining a next token if the node does not contain all child nodes or if the context pointer points back to a parent node.

associated set of attributes, and the hierarchical (nesting) rules of HTML tags. That is, a DTD component according to the present invention will declare, in the case of HTML for example, a tag set expression for paragraph text (e.g., "<p> . . . text stream . . . <p>"). Other expressions will be defined by such a grammar construct as provided by DTD components. Such definitions will be immediately understood by those skilled in the art. Once again, by separating the DTD components 106 from other components in the parser engine, it becomes possible to use the same system to parse a much wider range of document types and those containing expressions corresponding to different and varying rules of grammar. Simply put, this means that the same parser can provide an input to the browser biased (via the DTD components 106) to behave like any other HTML browser. The same can be said for XML, etc.

⁶ [0029] The present invention may be configured to allow for malformed documents and expressions to be constructed based on corresponding grammars contained within DTD components. That is, the present invention allows dynamic configuration of parsing systems through use of replaceable DTD components 106, and the transformation of otherwise malformed documents and expressions into well-formed expressions that a content model, for example, can later understand and process. Such transformation processes may be borrowed from artificial intelligence processing schemes and may involve rules of propagation, etc. For example suppose a parsing engine 100 were to realize a token for the start of paragraph text outside of an HTML file starting tag. Such a construct would otherwise be inappropriate and, in some cases by some browsers, unrenderable and therefore discarded. In contrast, the present invention now allows such tokens and constructs to be "fixed" or corrected (i.e., transformed into well-formed expressions) based on document context as defined by replaceable DTD components provided by the present invention, etc. That is, by recognizing the context of the tokens realized by a parsing engine provided by the present invention, etc., the present invention can transform otherwise malformed expressions and documents into well-formed objects which can be processed by an appropriate content model.

The “errors” corrected by Applicants’ claimed invention are grammatical and syntactical errors, while the errors described in Gessner are “...network traffic stoppages or delays, etc.”⁷

3. Error recovery means, which corrects the apparent error in accordance with the predetermined rule. Gessner does not teach or suggest the claimed error recovery means. Applicants’ amended claims recite a set of syntax recovery units that individually employ simple functions for correcting specific types of errors. Specifically, the claimed error recovery means is claimed to selectively employ the syntax recovery means based on the error type in accordance with predetermined rule in order to correct a variety of errors in the data string.

This is claimed to be done by a method that comprises sequentially using syntax recovery units where a syntax recovery unit receives an error message if a token does not grammatically match the syntax pointer, and the syntax recovery unit sequentially passes the error to a subsequent syntax recovery unit until a syntax recovery unit matching the error type is found.

Moreover, Applicants’ syntax recovery unit is claimed to:

- a) reset the current abstract syntax tree, buffer, and context pointer;
- b) recover a pointer where in accordance with said syntax rules no error exists; and
- c) notify the parser that the correction was successfully applied.

By way of contrast, while Gessner initially calls for implementing “robust error handling”⁸. However, while Gessner calls for some kind of “robust error handling”, the

⁷ [0037] A third phase of operation involves tokenization. The tokenizer aspect of parser component 104 controls and coordinates the tokenization of the input stream into a collection of tokens. Different grammars will have their own subclasses of tokens as well as their own corresponding DTD components. As the tokenizer runs, it repeatedly calls methods to get additional tokens for processing. Such tokenization iteration continues until an end-of-file occurs on an input stream, an unrecoverable error occurs such as network traffic stoppages or delays, etc.

only error handling that Gessner describes is directed to "...network traffic stoppages or delays, etc." and not to the errors disclosed by Applicants.

Gessner fails as an anticipatory reference and neither teaches nor suggests the claimed elements of Applicants' inventions.

⁸ [0023] A parsing engine such as parsing engine 100 represents a first stage in a sequence of system operations that interact in order for a network client or web browser to display and manifest HTML and other types of documents. In order for a layout engine to successfully process content received via a network connection, for example, parsing engine must be fast, extensible, and above all, it must offer robust error handling. The parsing engine in the context of the particular invention, and, in particular, parsing engine 1000 has a modular design that actually permits a system to parse almost any kind of data. However, like any web browser, the present invention's parsing engine 100, in particular, is optimized for HTML and other such markup languages. Conceptually, a parsing engine like parsing engine 100 is used to transform a source document from one form to another. In the case of HTML, for example, parsing engine 100 transforms the hierarchy of HTML tags (the source form) into another form that an underlying layout and display engine requires for appropriate rendition (the target form) based on a particular designed content model.

Conclusion

Based on the above discussion, it is respectfully submitted that the pending claims describe an invention that is properly allowable to the Applicants.

If any issues remain unresolved despite the present amendment, the Examiner is requested to telephone Applicants' Attorney at the telephone number shown below to arrange for a telephonic interview before issuing another Office Action.

Applicants would like to take this opportunity to thank the Examiner for a thorough and competent examination and for courtesies extended to Applicants' Attorney.

Respectfully Submitted


Certificate of Mailing

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on the date shown below with sufficient postage as Certified Priority Mail (Certified Label 7004 1160 0003 1436 5539) in an envelope addressed to the Commissioner for Patents, Mail Stop NF, PO Box 1450 Alexandria Virginia, 22313-1450

Date of deposit: August 18, 2004

Person mailing paper: Richard M. Goldman

Signature: 


Richard M. Goldman, Reg. # 25,585
371 Elan Village Lane, Suite 208
San Jose, CA 95134
Voice: 408-324-0716
Fax: 408-324-0672
E-mail: goldmanptn@aol.com